(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
12 April 2001 (12.04.2001)

PCT

(10) International Publication Number
WO 01/26004 A2

(51) International Patent Classification[7]: G06F 17/60

(21) International Application Number: PCT/US00/27422

(22) International Filing Date: 4 October 2000 (04.10.2000)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
09/411,666 4 October 1999 (04.10.1999) US

(71) Applicant: KANA COMMUNICATIONS, INC. [US/US]; 740 Bay Road, Redwood City, CA 94063 (US).

(72) Inventors: KASRAWI, Nawwar; 2034 San Luis Avenue, #4, Mountain View, CA 94043 (US). KANERVA, Jonni; 1728 Pilgrim Ave., Mountain View, CA 94040 (US). AMBROSE, Craig, M.; 14471 Springer Ave., Saratoga, CA 95070 (US). PHELPS, William, R.; 244 Leland Ave., Menlo Park, CA 94025 (US).

(74) Agent: JOHANSEN, Dag; Stattler Johansen & Adeli LLP, P.O. Box 51860, Palo Alto, CA 94303-0728 (US).

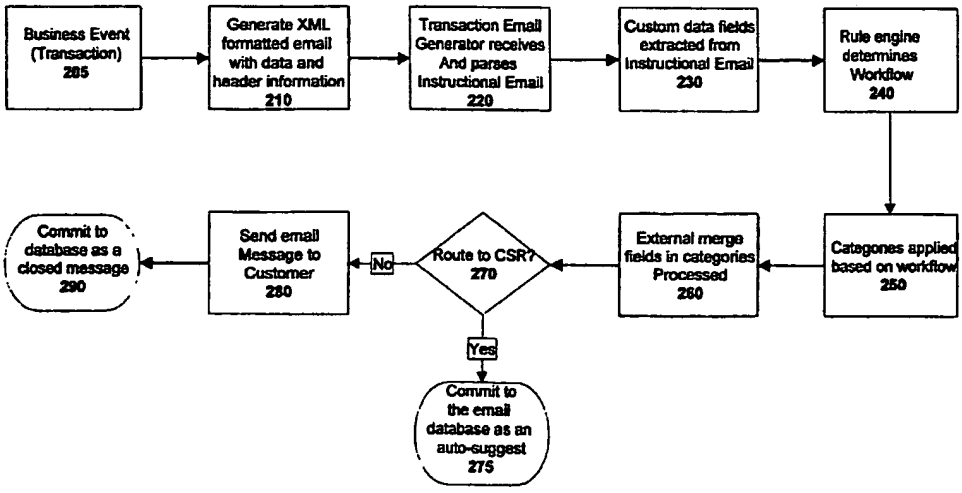
(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CR, CU, CZ, DE, DK, DM, DZ, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, UZ, VN, YU, ZA, ZW.

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published:
— *Without international search report and to be republished upon receipt of that report.*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

(54) Title: METHOD AND APPARATUS FOR INTERPROCESS MESSAGING AND ITS USE FOR AUTOMATICALLY GENERATING TRANSACTIONAL EMAIL



(57) Abstract: A method of defining and passing messages among separate computer application servers is disclosed. The method uses a specially formatted extensible markup language (XML) formatted message known as an extensible mail format (XMF). The XMF message includes attribute and data value pairs. The method is defined with reference to a method of creating transactional email messages. The method of creating transactional email messages receives specially formatted instructional email messages from other computer systems. The instruction email message includes a destination address, zero or more data fields, and zero or more instructions. The method parses the instructional email message to extract data from the data fields. The method then generates a template outgoing message from the instructional email message. The template outgoing message usually contains one or more merge fields. The method then fills in the merge fields using the data extracted from the instructional email message or external data sources to create a final outgoing message.

WO 01/26004 A2

Method and Apparatus For Interprocess Messaging And Its Use for Automatically Generating Transactional Email

FIELD OF THE INVENTION

The present invention relates to the field of interprocess messaging for computer systems. In particular the present invention discloses a method for performing interprocess messaging using email and its use in an application for automatically generating transactional email.

BACKGROUND OF THE INVENTION

The Internet has become a vast commercial marketplace. For example, Internet users may buy and sell financial instruments such as stocks and bonds, buy clothing, refinance a mortgage, or purchase and download commercial software.

The Internet based transactions usually occur between a consumer's client computer system and Web server run by a particular government or commercial entity. The Web server computer will typically communicate with an application server, which may in turn communicate with a number of other computer systems in order to complete a transaction. For example, the application server may check an inventory database to determine if a particular item is in stock, the application server may test a credit database computer to determine if a transaction with a particular customer should be accepted, and the application server may access an accounting computer to enter a new order. Each of these communication sessions between server programs requires the exchange of data. It would be desirable to have a simple method of passing data between various server applications.

One of tasks that the application server may perform when processing an Internet transaction is to create a transactional email message that will act as the virtual equivalent of a "paper trail" for the Internet transaction. Transactional email is defined as email sent between a company and a customer as a result of a business event or transaction. The transactional email will typically be highly personalized and might contain specific customer information such as account or order status.

Many companies are increasingly communicating proactively with customers during the customer interaction process. For example, an Internet commerce web site may send a transactional email confirming an order or notifying a customer of the status of an order throughout the order's lifecycle. Similarly, an Internet based brokerage house may send a transactional email confirming each trade made by a client. A package transport company may wish to send a transactional email stating when a package was accepted and another transactional email when the package was delivered. All of these transactional email examples may require support from many different inflexible legacy systems such as mainframes, accounting programs, trading programs, etc. To simplify the process of generating transactional email, it would be desirable to have a flexible automatic transactional email system.

SUMMARY OF THE INVENTION

A method of defining and passing messages among separate computer application servers is disclosed. The method is defined with reference to a method of creating transactional email messages. The method of creating transactional email messages receives specially formatted instructional email messages from other computer systems. The instruction email message includes a destination address, zero or more data fields, and zero or more instructions. The method parses the instructional email message to extract data from the data fields. The method then generates a template outgoing message from the instructional email message. The template outgoing message usually contains one or more merge fields. The method then fills in the merge fields using the data extracted from the instructional email message or external data sources to create a final outgoing message.

Other objects, features, and advantages of present invention will be apparent from the company drawings and from the following detailed description.

BRIEF DESCRIPTION OF THE DRAWINGS

The objects, features, and advantages of the present invention will be apparent to one skilled in the art, in view of the following detailed description in which:

Figure 1 illustrates a typical computer environment that may use the automated transactional email teachings of the present invention.

Figure 2 illustrates a flow diagram setting forth one embodiment's method of generating transactional email messages.

Figure 3A illustrates a first example embodiment of an Extensible Message Format (XMF) formatted instructional email message for creating a transaction email message.

Figure 3B illustrates a second example embodiment of an Extensible Message Format (XMF) formatted instructional email message that uses the "from:" field to designate an intended transactional message recipient.

Figure 4 illustrates an example template message body for a normal order confirmation message.

Figure 5 illustrates an example template message body for a gold customer order confirmation message.

Figure 6 illustrates an example final outgoing message built from the template message body of **Figure 4**.

Figure 7 illustrates an example final outgoing message built from the template message body of **Figure 5**.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

A method and apparatus for interprocess messaging is disclosed. In the following description, for purposes of explanation, specific nomenclature is set forth to provide a thorough understanding of the present invention. However, it will be apparent to one skilled in the art that these specific details are not required in order to practice the present invention. For example, the present invention has been described with reference to a system for automatically generating transactional email. However, the same techniques can easily be employed for other types of interprocess messaging.

Network Systems

Figure 1 illustrates a typical local area network (LAN) environment **170** at a commercial company. The local area network (LAN) **170** of **Figure 1** includes a number of server systems for performing various computational chores. For example, the sample embodiment of **Figure 1** includes an email server **171** for distributing email messages, an accounting server **173** for maintaining financial records, a database server **175** for storing data, and a World Wide Web (WWW) server **177** for storing and distributing web pages. Note that any two servers may generally reside on either the same physical computer or distinct computer systems.

The LAN **170** further includes client computer systems that can access and modify the information in the various server systems. **Figure 1** illustrates two client computer systems **125** and **135** coupled to the LAN **170** (although many more clients may be present). In the embodiment of **Figure 1**, telephone terminals **121** and **131** accompany the client computer systems **125** and **135** forming workstations **120** and **130**. Telephone terminals **121** and **131** may be coupled to a Private Branch eXchange (PBX) that is coupled to the Public Switched Telephone Network (PSTN). In this manner, employees at workstations **120** and **130** may converse with customers over the telephone while accessing server resources using client computer systems **125** and **135**.

Note that some of the servers on LAN **170** are also coupled to the global Internet **100**. For example, email server **171** and a World Wide Web (WWW) server 177 are coupled to

both LAN **170** and Internet **100**. In this manner, customers or potential customers may contact the company through the Internet **100** without the need for human interaction.

A number of different inter-server transactions may take place in the environment of **Figure 1**. For example, a customer may call a customer service agent seated at workstation **120** and initiate an order for a product such that the customer service agent enters an order into workstation, which communicates the order to an order management server **176**. The order management server **176** then stores a record of the order in the database server **175**. Alternatively, a customer may place an order for a particular product by filling in a form on the Internet web site hosted by WWW server **177**. Once an order has been placed, the order management server **176** may access a credit database on another server (not shown) and determine if the customer has sufficient credit for the order. If the customer is approved, the order management server **176** may then need to enter financial information into the accounting server **173**.

At various stages, the company may wish to send its customers email messages about the status of their orders. To help create such email messages, the company may employ a transactional email program. In the embodiment of **Figure 1**, the transactional email program is running on transactional email server **179**. However, the transactional email program may run on any suitable server system. The transactional email program may generate an email message that is sent out to the customer using email server **171**.

A interprocess messaging system is needed to handle communications between servers, which may exist on distinct physical computers. The application layer for communications protocols is defined in the Transmission Control Protocol/Internet Protocol (TCP/IP) reference model as the layer on top of the transport layer, which typically utilizes TCP as a communications protocol. The application layer sits at the top of communication layers, directly below the sending and receiving processes, which may run on different servers. There are several methods to provide application layer communications. These methods fall into one of two categories:

- Proprietary protocols
- Standard distributed object protocols

Proprietary application-level protocols are developed and used by companies to communicate information either among internal information systems or among information systems using that company's commercially available software. In either case, the details of the protocol are not made generally available. For example, a company may develop a proprietary application-layer protocol using Java object serialization on sockets to create remote procedure calls (RPCs).

Standard distributed object protocols allow objects to be accessed in distributed environments. These protocols include Java Remote Method Invocation (RMI), Distributed Component Object Model (DCOM), Common Object Request Broker Architecture (CORBA), and Enterprise Java Beans (EJB).

Above the application layer are the sending and receiving processes. These processes include domain-specific applications such as accounting software, order management software, or inventory control software. The domain-specific applications may call standard distributed object protocols directly. messaging systems such Alternatively, domain-specific applications may use commercially available messaging systems such as BEA TUXEDO, IBM MQSeries, and Tibco TIB/Rendezvous. Messaging systems such as these may use either proprietary protocols or standard distributed object protocols.

Both proprietary application-layer protocols and standard distributed object protocols allow a first program to make a function call into another program that is running on the same server or on another server. Though useful, these application layer protocols are often operating systems specific. Thus, a function call made from a first program running on a first server running a first operating system may not work properly with a second program running on a second server running a second operating system that is not the same as the first operating system. Furthermore, various network topology, routing, and firewall issues may prevent a particular program from making a function call into a second program on a second server. It would therefore be desirable to have another method for performing interprocess messaging.

Commercially available messaging systems have a similar limitation in that they must be integrated with both the first program and second program. Typically in a large commercial entity, enterprises may standardize on or more messaging systems. In order for disparate servers to communicate however, they must use the same messaging protocol. Thus,

various subsets of servers may communicate with each other, but full interconnectivity of servers is rare.

The present invention overcomes the application-layer protocol and higher-level messaging compatibility issues by using a combination of standard, protocols. The present invention uses the Simple Mail Transport Protocol (SMTP) to accomplish application-layer communications. The invention uses SMTP servers and clients as the sending and receiving processes. Virtually all server-class computing systems can function as both an SMTP server and an SMTP client. Because of the ubiquity of SMTP, there is an abundance of publicly available documentation on SMTP. Accordingly, corporate management information system (MIS) departments are typically extremely familiar with SMTP. Thus, there is little resistance within an organization's MIS department with supporting SMTP on any application server.

The present invention further leverages standard protocols by using XML as a message format. This messaging format is discussed in the next section. The novelty of the present invention is that although it uses standard STMP and XML protocols, it combines them in a novel manner to create a messaging system that can be universally implemented and supported so that nearly any two computer systems linked by a LAN or the Internet can communicate.

Extensible Email Format (XMF)

To provide a simple, flexible, and ubiquitous method of performing interprocess messaging, the present invention introduces an eXtensible Message Format (XMF). The eXtensible Message Format (XMF) is used to encode interprocess messaging data into email messages. In one embodiment, the eXtensible Message Format (XMF) email message comprises a Multipurpose Internet Mail Extension (MIME) formatted message that contains a special header variable identifying it as an XMF message. The email message is a standard Internet mail message format as defined by the Internet Engineering Task Force (IETF) Request for Comments (RFC) number 822 (RFC 822). The Multipurpose Internet Mail Extension (MIME) format is defined by IETF RFCs 2045 through 2049. The eXtensible Message Format (XMF) further includes a text/xml segment that contains interprocess messaging data fields that have been encoded using eXtensible Markup Language (XML).

In an alternative embodiment of the present invention, the eXtensible Message Format (XMF) email message comprises a Secure Multipurpose Internet Mail Extension (S/MIME) formatted message. The S/MIME specification consists of two documents: S/MIME Message Specification and S/MIME Certificate Handling. Both of these documents are Internet Drafts that have been submitted to the IETF for approval as a standard.

S/MIME is a specification for secure electronic mail. S/MIME was designed to add security to email messages in MIME format. The security services offered are authentication using digital signatures and privacy using encryption.

The email transport mechanism was chosen as a data exchange format since email is a ubiquitous, well-understood, and robust data format. The use of email as a data exchange transport system allows a Local Area Network (LAN) to function as a unified message-passing based multi-processor computer system. Furthermore, the use of email as a data exchange transport system allows virtually any computer with a link to the global Internet to be used within a multiprocessing application. This is generally true since email is widely available to almost any computer system linked to the Internet even if the computer system is shielded by a packet-filtering router, a proxy based firewall, or any other security system. Thus, any computer system that can create and send an XML-formatted email message can create an eXtensible Message Format (XMF) formatted email message that can be used for interprocess messaging. The XMF specification has been formally documented such that any programmer with such documentation may be able to write code that directly generates interprocess messaging email in XMF.

XMF allows an email message that contains any number of custom data fields to be sent to any other computer program that can receive email. These data fields are configured as part of a database that is dynamically extensible. The data fields are available as attributes and values for the system that receives the XMF mail.

The data fields of XMF messages may comprise complex expressions that need to be parsed and processed by the receiving program. For example, certain fields may refer to data outside of the receiving program and XMF message. One specific method of referring to external data is to specify a Uniform Resource Locator (URL) that refers to another server. The URL may be constructed with nested sub expressions that also need to be evaluated. After a

URL expression data field is processed and resolved, the receiving program opens a socket to the URL and fetches any data that is provided. The receiving program may then use the fetched data. Thus, a receiving program that receives an XMF message containing a URL that points to other data will use the URL to retrieve the data from the computer system referred to in the URL. By using a URL, the data in the external referrals may also use an easily defined, ubiquitous, and well-implemented data retrieval mechanism. For example, the URL may refer to data accessible using the well-known HyperText Transport Protocol (HTTP).

The use of XMF as a messaging format and email as a interprocess messaging transport protocol allows otherwise decoupled systems to communicate and share data. Thus, any two programs may communicate with each other using email and XMF. For example, referring to **Figure 1**, any programs running on the computers coupled to the LAN **170** such as systems **171**, **173**, **175**, **176**, **177**, **179**, **125**, and **135** can easily communicate with each other using email and XMF. Furthermore, those computer systems may also communicate data with computers on the Internet **100** (such as Internet Server system **111** or Internet client systems **110** and **112**) provided that the router or firewall that connects LAN **170** to the Internet **100** allows email to pass through.

For example, application server **105** may create the informational XMF message to initiate a transaction. In **Figure 1**, the application server **105** uses a standard email protocol such as the Simple Mail Transport Protocol (SMTP) to send an informational XMF email to transactional email server application **179**. Because of the ubiquity of email protocols such as SMTP, the use of XMF and email for communications becomes a general-purpose method for conducting transactions.

Note that for security reasons, the application server **105** may want to utilize digital authentication and encryption particularly if the receiving system (transactional email server application **179**) is accessible from the Internet. Using commercially available digital authentication and encryption technologies, many of which are based on the S/MIME standard, application server **105** may encrypt and/or digitally sign the XMF email message. T he transactional email server application **179** can verify that the email message was sent from a trusted source. The transactional email server application **179** will recognize the digital authentication in order to accept and process the XMF email. Furthermore, without the correct

key, the transactional email server application will not be able to read the message. Thus, the application server can guarantee that the message will be understandable only by trusted destinations **179**.

An XMF message consists of a header and a body. The XMF header contains information on the origin, destination, and type of the message. The body contains the attribute-value pairs that constitute the primary information in the message.

XMF Header Format

An XMF header may have the following six components:

Name	Explanation
From	Standard email header (RFC 822)
To	Standard email header (RFC 822)
Subject	Standard email header (RFC 822)
MIME-Version	Standard MIME header (RFCs 2045-2049)
Content-Type	Standard MIME header (RFCs 2045-2049)

Here is an example XMF header, from an XMF message that was generated from customer input to a web form:

```
From: forms@acme.com
To: support@acme.com
Subject: Customer Request Form
MIME-Version: 1.0
Content-Type: "multipart/alternative; boundary=1234567890"
```

XMF Body Format

An XMF message, as previously set forth, is encapsulated in a standard MIME email message. Most email applications can read a plain text and/or an HTML version of the email message. For completeness, an XMF message may contain a human-readable version of the transactional message as either plain text or HTML. The XMF specification goes one step further and takes name-value pairs and encodes them allowing for processing by the receiving server. The encoded portion of an XMF message is structured using eXtensible Markup Language (XML). Specifically, XMF messages are encoded in XML using a document type definition known as Message Markup Language (MML). The MML specification is readily available for implementers to allow them to build applications around XMF.

EXtensible Markup Language (XML) is a meta-language for defining other more specific markup languages. XML provides a facility to define tags and the values and the relationships among them. Message Markup Language (MML) is a simple markup language defined using XML. In fact, XMF is defined by the existence of an MML segment within a multipart alternative MIME email message. When the processing service detects the presence of an MML segment the message is handled in a special way.

The following code provides and example of a transactional message described using MML:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<Root>
  <Message>
    <Attribute name="Message Type">FormMessage</Attribute>
    <Attribute name="First Name">Sandy</Attribute>
    <Attribute name="Last Name">Jones</Attribute>
    <Attribute name=
          "Email Address">sandy@mumblefoo.com</Attribute>
    <Attribute name="Phone Number">650-123-4567</Attribute>
    <Attribute name="Request">Send New Password</Attribute>
    <Attribute name=
          "Subject">FM - Lost Password</Attribute>
  </Message>
</Root>
```

The first line describes the MML document, including the XML version number and language encoding type. The XML encoding-type should be specified but is defaulted to the ISO-8859-1 specification. In a current embodiment, the supported encoding-types in MML are "ISO-8859-1" and "US-ASCII".

The message is then made up of named attributes and their associated data values wrapped within a message. These elements are encapsulated at the root level of the XML document since all elements of an XML document must be maintained within the root. The named attributes can be anything, but for a transactional message the attributes will describe the instructions and values that make up the transaction. In the present invention, the attribute with the name "Message Type" is used to determine the type or purpose of the message. In general, the attribute names should match up with well-known data fields within the receiving system.

The attribute names are limited to only upper and lower case letters, numeric digits, hyphens, or underscores, and must begin with a letter and end with a letter or digit.

The values associated with an attribute normally contain legal printable and white space characters that are available within the document's specified encoding. Further, four characters ('<', '&', '"', and '>') have special significance to XML and hence MML and must be encoded as entities rather than as plain text. The single quote character (') may optionally also be encoded as an entity.

The following table describes how various special XML characters may be entered into XMF messages:

Character	ASCII	Entity
Left Angle Bracket	60	`<`
Ampersand	38	`&`
Double Quote	34	`&quot;`
Right Angle Bracket	62	`>`
Single Quote (opt.)	39	`&apos;`

As mentioned, MML is a simple, yet well-defined XML definition for describing transaction messages. For reference purposes, the following code defines the XML Document Type Definition (DTD) for MML:

```
<!DOCTYPE Root [
 <!ELEMENT Root (Message+)>
 <!ELEMENT Message (Attribute)+>
 <!ELEMENT Attribute (#PCDATA)>
 <!ATTLIST Attribute name CDATA #REQUIRED>
]>
```

The Document Type Definition (DTD) is not a required component of an XMF message but is useful in programs that validate the structure of the MML segment when the message is being constructed.

When the receiving process, such as the transactional email program on transactional email server **179** in **Figure 1**, detects a XMF message, the MML portion is run through an XML interpreter and the attributes and data values are analyzed. Most of the attributes will be stored in the database **175** if the attribute names correspond to standard data

elements or custom-defined ones. Some of these values can dictate that the receiving process performs an action. This will always be the case for transactional messages and the action performed will be predicated by the values that are also contained in the XMF message. In certain cases the values in the message will indicate that data should be retrieved from another source to be included in the response to the transaction message.

XMF Text and HTML Body Version Formats

The text portion may contain a plain text version of the same information in the MML portion. Here is an example of the plain text part:

```
Lost Password:

First Name:  Sandy
Last Name:  Jones
Email Address:  sandy@mumblefoo.com
Phone Number:  650-123-4567
Request:  Send New Password
Subject:  FM - Lost Password
```

Similarly, the HTML portion may contain an HTML formatted version of the same information in the MML portion. Here is an example of an HTML part:

```
<html><head>
<META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=iso-
8859-1">
<title>Lost Password</title></head><body>
<br><h1>Lost Password</h1>
<table>
  <tr><td>First Name:</td><td>Sandy</td></tr>
  <tr><td>Last Name:</td><td>Jones</td></tr>
  <tr><td>Email Address:</td><td>sandy@mumblefoo.com</td></tr>
  <tr><td>Phone Number:</td><td>650-123-4567</td></tr>
  <tr><td>Request:</td><td>Send New Password</td></tr>
  <tr><td>Subject:</td><td>FM - Lost Password</td></tr>
</table><br><br><h1> End of Form Data </h1>
</body></html>
```

It is usually desirable to replace the XML special characters within fields with their entity/escaped forms in the HTML document, as well as in the XML document.

XMF Body Semantics

The XMF body may contain a plain text part, an HTML part, and a MML part, as previously set forth. The plain text part of the message may be used to display the message to simple clients. A more complex client may display the HTML version of the XMF message. If the HTML part of the message is not included, a simple client will display the plain text message and will include the plain text part in replies.

The MML part of the document is parsed by a receiver program in the present invention such that each of the name-value pairs may be submitted to a database server **175**. If the name matches a valid field name in the database couple to the database server **175**, the value is entered into the table with that field name. There are standard fields and custom fields. Standard attributes are predefined in the database server **175** and correspond to fields in a regular SMTP email message. In the present invention, custom fields may be added to the database system using the application server **105**. The following lists provides one set of standard fields that may be used:

Standard Field	**Definition**
First Name	The first name of the customer.
Last Name	The last name of the customer.
Full Name	The name of the customer
Email Address	The email address of the customer
Subject	The subject of the message.

An Example Application of XMF

The XMF format can be used to generate personalized email messages that are appropriate for particular business transactions that transpire. This section will describe how XMF may be used to generate such email messages.

Transactional Email

Transactional email is comprised of email sent between a company and a customer as a result of a business event or transaction. Transaction email messages are usually highly personalized and contain specific information related to the specific business transaction such as customer account information, order status, transaction-specific information, etc.

Many different business events may require transaction email messages to be sent. For example, an Internet commerce web site may send a transaction email to confirm an order or notify a customer of the status of an order throughout the order's lifecycle. Similarly, an Internet based brokerage house may send a transactional email message confirming a specific stock trade made by a client. These transactional email examples may require support from many different inflexible legacy systems such as mainframes, accounting programs, trading programs, etc. To simplify the process of generating transactional email, the present invention introduces a system for automatically generating transactional email messages. The system allows many different transactional email messages to be generated.

The automatic transactional email system of the present invention provides many benefits to its users. The content and formatting of transactional email can be defined in an easily manageable user interface using existing outgoing email response templates. The email response system receives an email including just the data and instructions needed to generate and send an outgoing response email message. The system allows the workflow associated with a specific business event to be abstracted away from the specific computer system reporting the business event. The transactional email system can decide what message to send based on information in the header and body of an instructional email message and defined rules. The generated transactional email may be stored in a database for future reference.

A Transactional Email Program

The goal of the transactional email program is to send transactional email based upon a specific transaction. The transactional email program must obtain the available transaction information from the systems involved in the transaction. Since each different system may internally represent information in different formats, each system needs to generate a translation of the information in XMF. Because the XMF specification is simple and well specified, the task of creating a translator program is very straightforward.

The transactional email program then uses the XMF formatted email message to create an appropriate personalized email message for the customer/potential customer. The transactional email program does not simply reformat the XMF formatted email information message. The transactional email program processes the XMF formatted email information

message and by accessing other servers, parses command codes, logs the message, and adds specific coding that will allow the company to handle any customer replies to the final transaction email message sent to the customer

Figure 1 illustrates a company LAN **170** that includes a transactional email server **179** for automatically generating transactional email. Any other computer system on the LAN **170** or on the Internet **100** can submit XMF messages that will generate transactional email messages. However, the transactional email server **179** should authenticate all incoming XMF messages to ensure that such messages come from an authorized source. In addition, the servers generating the XMF messages should digitally encrypt the messages, particularly if the messages are being sent over the Internet **100**. As previously set forth, this security can be accomplished using commercially available digital encryption and authentication.

Figure 2 illustrates a flow diagram that describes how the transactional email program works with other elements in a computer network to create, log, and send transactional email messages. Referring to step **205**, a transaction event begins the process. As previously set forth, the transaction event may be a customer purchasing a product by placing a telephone call to a customer service agent; a potential customer requesting information about a particular product by filling in a form on a web server; an accounting program determining that a particular customer is behind on payments; or any other transactional event that involves a computer system.

Next, at step **210**, an XMF formatted information email message is created and sent to the transactional email server **179**. The XMF formatted email message is referred to as an "instructional email." The XMF email message contains an email message of the customer/potential customer, zero or more pieces of data, zero or more Uniform Resource Locator links to other data, and instructional information such as whether the message should be sent directly to the customer/potential customer or routed to a queue for review by a customer service representative. The instruction information may be presented in keyword format. A rule engine will determine the appropriate workflow actions to take based upon the keywords. The workflow actions may include categorizing an email message, applying template text that forms the body of a final outgoing email message, storing the final outgoing email message, etc. In one embodiment, the instructions comprise extra information in the email message header field.

The transactional email program receives and parses the instruction email at step **220**. In most embodiments, the transactional email program should first authenticate the received XMF email to ensure that it came from a trusted source. At step **230**, the transactional email program extracts data fields in the structured XMF instructional email. In one embodiment, the transactional email program stores the extracted information in a database, such as the one accessed by database server **175**. The transactional email program will reference this extracted data as necessary.

Next, at step **240**, the transactional email program determines the workflow actions that are necessary. The transactional email program uses the various data fields, instructions, and header fields to determine the necessary workflow. In one embodiment, the transactional email program may use an incoming email rule-processing engine to evaluate the instructional email. On example of an email rule-processing engine is defined in the U.S. patent application " Method And Apparatus For Performing Enterprise Email Management" filed on November 17,1998 and having Serial number 09/193,285. At step **250**, the transactional email program applies "categories" to the instruction email message. Category templates are used to create the body of an outgoing message. The template body may include one or more merge fields that need to be filled in.

At step **260**, the transactional email program fills in the merge fields of the outgoing message template body. Many of the fields may simply be filled in with data extracted during the extraction step **230**. Other merge fields contain complex expressions that must be parsed and processed. For example, some fields may contain merge fields that contain URLs that refer to externally accessible data. The transactional email program resolves the URLs, fetches the data, and merges the fetched data into the message field.

At step **270**, the transactional email program determines how the final generated outgoing message should be handled. The transactional email program examines an instruction or data field in the instructional email to determine if the message should be sent out directly or routed to a customer service representative (CSR) queue. The CSR queue will provide the message to a customer service representative for inspection and confirmation before the message is sent out.

If the message is to be inspected before delivery, then the message is routed to CSR as an automatically generated suggested message at step **275**. The CSR examines the suggested message. If the CSR approves of the final outgoing message, the message is sent to the customer. The CSR may modify the message before sending. After the message has been approved or modified and subsequently approved, the final outgoing message may be logged into a database.

Referring back to step **270**, if the finalized message does not need to be inspected before delivery, then the final outgoing message is sent directly to the customer at step **280**. The sent final outgoing message may be logged into a database at step **290**.

An Example Transaction Email XMF transaction

To best describe one embodiment of the transactional email system of the present invention, a detailed example is provided. **Figure 3A** illustrates one embodiment of an example XMF formatted instructional email message. The example XMF formatted instructional email message of **Figure 3A** is generated by any computer system in response to a business transaction such as a customer order.

The instructional email message of **Figure 3A** may be sent by an ecommerce web site or from a client program run by a telephone sales agent sitting at a computer workstation such as workstations **120** and **130** in **Figure 1**.

In an alternate embodiment, the instructional message may be formatted to appear as if was sent from the customer to communicate with. For example, the instructional email message of **Figure 3B** appears to be sent from joe@public.com by having the message origination source set the "from field" of the instructional email address as the destination customer address. In this manner, any automated incoming email processing system may be used to respond to the message. Thus, the teachings from in the U.S. patent application "Method And Apparatus For Performing Enterprise Email Management" filed on November 17,1998 and having Serial number 09/193,285 can be used.

The instructional email message of **Figure 3A** or **3B** is sent to a transactional email program. The transactional email program parses the instructional email message. In the

example instructional email message of **Figure 3**, the following data fields are extracted from the XML body:

- Message Type = OrderStatus
- Email Address = joe@public.com
- Account Number = 1234567
- Order description = http://customer/order?
- Order number = http://customer/ordernumber?
- Order status = http://customer/status?

To process instructional email messages, several transactional email processing rules are configured. For the specific instructional email messages of **Figures 3A** and **3B**, a couple of specific transactional email processing rules have been configured.

In the examples of **Figures 3A** and **3B**, the rules uses "instructions" embedded in the message header information. One of the rules looks for "X-CustomerEvent-OrderConfirmation" in the message header, and autoresponds to such messages as an outgoing customer order confirmation. In one embodiment, the rule acts by categorizing the instructional email message as an outgoing confirmation message. The transactional email program then creates an outgoing email message using an outgoing customer order confirmation template associated with the outgoing confirmation message category.

Another rules additionally looks for "X-CustomerEvent-GoldCustomer" and autoresponds to the message a specific gold customer confirmation message. Again, in one embodiment the rule acts by categorizing the instructional email message such that the transactional email program creates an outgoing email message using an outgoing customer order gold customer confirmation template associated with the category.

The two rules are defined as follows:

```
header.contains("X-CustomerEvent-OrderConfirmation") and header.contains("X-
        CustomerEvent-Gold") -> autorespond(OrderConfirmation, Gold)
header.contains("X-CustomerEvent-OrderConfirmation") ->
        autorespond(OrderConfirmation)
```

where the rule engine executes the first rule that is satisfied. **Figure 4** illustrates how the template for the normal order confirmation category may appear. **Figure 5** illustrates how the template for the gold customer order confirmation category may appear.

example instructional email message of **Figure 3**, the following data fields are extracted from the XML body:

- Message Type = OrderStatus
- Email Address = joe@public.com
- Account Number = 1234567
- Order description = http://customer/order?
- Order number = http://customer/ordernumber?
- Order status = http://customer/status?

To process instructional email messages, several transactional email processing rules are configured. For the specific instructional email messages of **Figures 3A** and **3B**, a couple of specific transactional email processing rules have been configured.

In the examples of **Figures 3A** and **3B**, the rules uses "instructions" embedded in the message header information. One of the rules looks for "X-CustomerEvent-OrderConfirmation" in the message header, and autoresponds to such messages as an outgoing customer order confirmation. In one embodiment, the rule acts by categorizing the instructional email message as an outgoing confirmation message. The transactional email program then creates an outgoing email message using an outgoing customer order confirmation template associated with the outgoing confirmation message category.

Another rules additionally looks for "X-CustomerEvent-GoldCustomer" and autoresponds to the message a specific gold customer confirmation message. Again, in one embodiment the rule acts by categorizing the instructional email message such that the transactional email program creates an outgoing email message using an outgoing customer order gold customer confirmation template associated with the category.

The two rules are defined as follows:

```
header.contains("X-CustomerEvent-OrderConfirmation") and header.contains("X-
    CustomerEvent-Gold") -> autorespond(OrderConfirmation, Gold)
header.contains("X-CustomerEvent-OrderConfirmation") ->
    autorespond(OrderConfirmation)
```

where the rule engine executes the first rule that is satisfied. **Figure 4** illustrates how the template for the normal order confirmation category may appear. **Figure 5** illustrates how the template for the gold customer order confirmation category may appear.

When the transactional email server uses the templates of **Figure 4** and **Figure 5**, the merge fields in the templates are processed and resolved. Since these merge fields ultimately contain URLs, those URLs will be resolved, and the text returned by the URLs will be inserted. For example, the `<merge url="<merge name=order_description>account_number=<merge name="account_number">">` merge field of **Figure 4** will be processed into "= http://customer/order?account_number=1234567". This URL will then be resolved such that a data value is fetched from the URL. The data value is then placed into the merge field such that the message says "`Beanie Baby`" (the data retrieved after resolving the URL http://customer/order?account_number=1234567). In the example of **Figure 4**, the "last=true" attribute and value specify that the last item in the should be retrieved.

Figure 6 illustrates how the final outgoing message for the normal order confirmation category may appear. **Figure 7** illustrates how the final outgoing message for the gold customer order confirmation category may appear.

The foregoing has described a method and apparatus for automatically generating transactional email. It is contemplated that changes and modifications may be made by one of ordinary skill in the art, to the materials and arrangements of elements of the present invention without departing from the scope of the invention.

CLAIMS

We claim:

1. A method of distributing interprocess communication, said method comprising:

formatting an email message with extensible markup language in a first computer system;

encoding at least one attribute and data value pair in said email message; and

sending said email message to a second computer system to pass said at least one attribute and data value pair.

2. The method as claimed in claim 1, said method further comprising:

receiving said email message in said second computer system; and

extracting said at least one attribute and data value pair within said second computer system.

3. The method as claimed in claim 1 wherein said sending is performed using the Simple Mail Transport Protocol.

4. The method as claimed in claim 1, said method further comprising:

encrypting said email message in said first computer system before sending said email message.

5. The method as claimed in claim 4 wherein said encrypting is performed using the Secure Multipurpose Internet Mail Extensions.

6. The method as claimed in claim 1, said method further comprising:

digitally signing said email message in said first computer system before sending said email message.

7. The method as claimed in claim 1 wherein said email message passes through a firewall system.

8. The method as claimed in claim 1, said method further comprising:
encoding at least one instruction in said email message.

9. The method as claimed in claim 8 wherein said instruction comprises a message header field.

10. The method as claimed in claim 1 wherein said data value comprises an expression that must be evaluated.

11. The method as claimed in claim 1 wherein said data value comprises a Uniform Resource Locator.

12. An apparatus for distributing interprocess communication, said apparatus comprising:
a computer network, said computer network comprising more than one computer system; and
a first computer system, said first computer system formatting an email message with extensible markup language, encoding at least one attribute and data value pair in said email message, and sending said email message to a second computer system to pass said at least one attribute and data value pair;

13. The apparatus as claimed in claim 12, said apparatus further comprising:
a second computer system, said second computer system receiving said email message in said second computer system and extracting said at least one attribute and data value pair within said second computer system.

14. The apparatus as claimed in claim 12 wherein said first computer system sends said email message using the Simple Mail Transport Protocol.

15. The apparatus as claimed in claim 12 wherein said first computer system encrypts said email message before sending said email message

16. The apparatus as claimed in claim 15 wherein said first computer system encrypts using the Secure Multipurpose Internet Mail Extensions.

17. The apparatus as claimed in claim 12 wherein said first computer system digitally signs said email message before sending said email message

18. The apparatus as claimed in claim 12 further comprising:
a firewall system coupled to said computer network, said firewall system protecting said computer network from the Global Internet, said first computer on a first side of said firewall system;
wherein said second computer system is on a second side of said firewall system.

19. The apparatus as claimed in claim 12 wherein said first computer system encodes at least one instruction in said email message.

20. The apparatus as claimed in claim 19 wherein said instruction is encoded with a message header.

21. The apparatus as claimed in claim 12 wherein said data value comprises an expression that must be evaluated.

22. The method as claimed in claim 12 wherein said data value comprises a Uniform Resource Locator.

23. A method of creating transactional email messages, said method comprising:

accepting an instructional email message, said instruction email message comprising one or more data fields;

parsing said instructional email message to extract data from said data fields; and

generating a template outgoing message from said instructional email message, said template outgoing message containing merge fields; and

filling in said merge fields using said data from said data fields to create a final outgoing message.

24. The method as claimed in claim 23, said method further comprising:

sending said final outgoing message to a customer identified in said instructional email message.

25. The method as claimed in claim 23, said method further comprising:

sending said final outgoing message to a customer service representative.

26. The method as claimed in claim 23 wherein generating a template outgoing message from said instructional email message comprises processing said instructional

email message with a rule processing engine to determine a message category and generating said template outgoing message based upon said category.

27. The method as claimed in claim 23 wherein said instructional email message comprises an XML formatted message.

28. The method as claimed in claim 23, said method further comprising:
processing said merge fields in said template outgoing message by evaluating an expression.

29. The method as claimed in claim 28 wherein processing said merge fields further comprises:
resolving a URL.

30. The method as claimed in claim 23, said method further comprising:
resolving a URL in a merge field in said template outgoing message.

31. The method as claimed in claim 23 wherein said sending is performed using the Simple Mail Transport Protocol.

32. The method as claimed in claim 23 further comprising:
Authenticating said instructional email message.

33. The apparatus as claimed in claim 32 wherein said authenticating uses digital signatures.

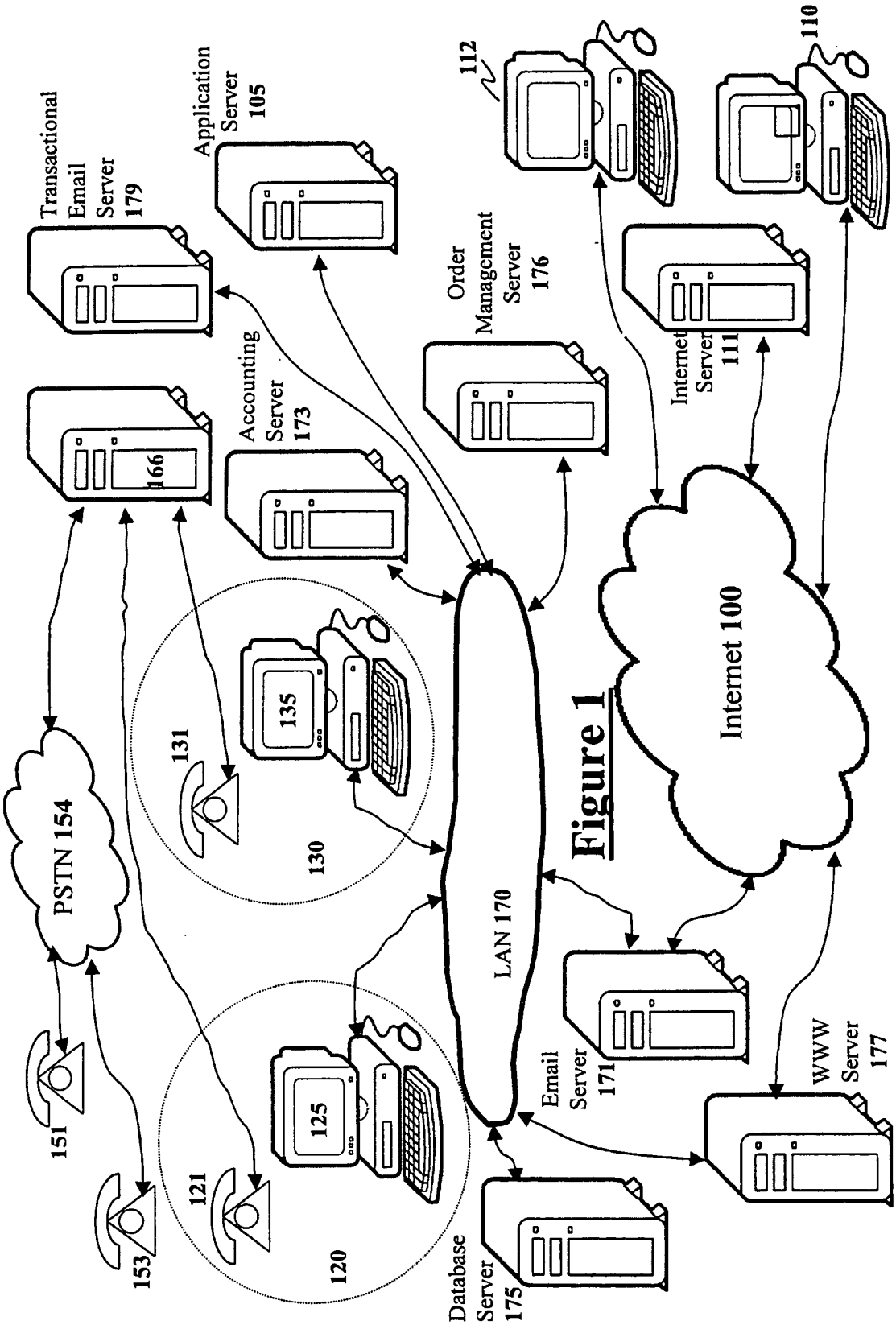

Transactional Email Server 179
Application Server 105
112
110
Order Management Server 176
Internet Server 111
166
Accounting Server 173
PSTN 154
131
135
130
LAN 170
Internet 100
151
153
121
125
120
Email Server 171
WWW Server 177
Database Server 175

Figure 1

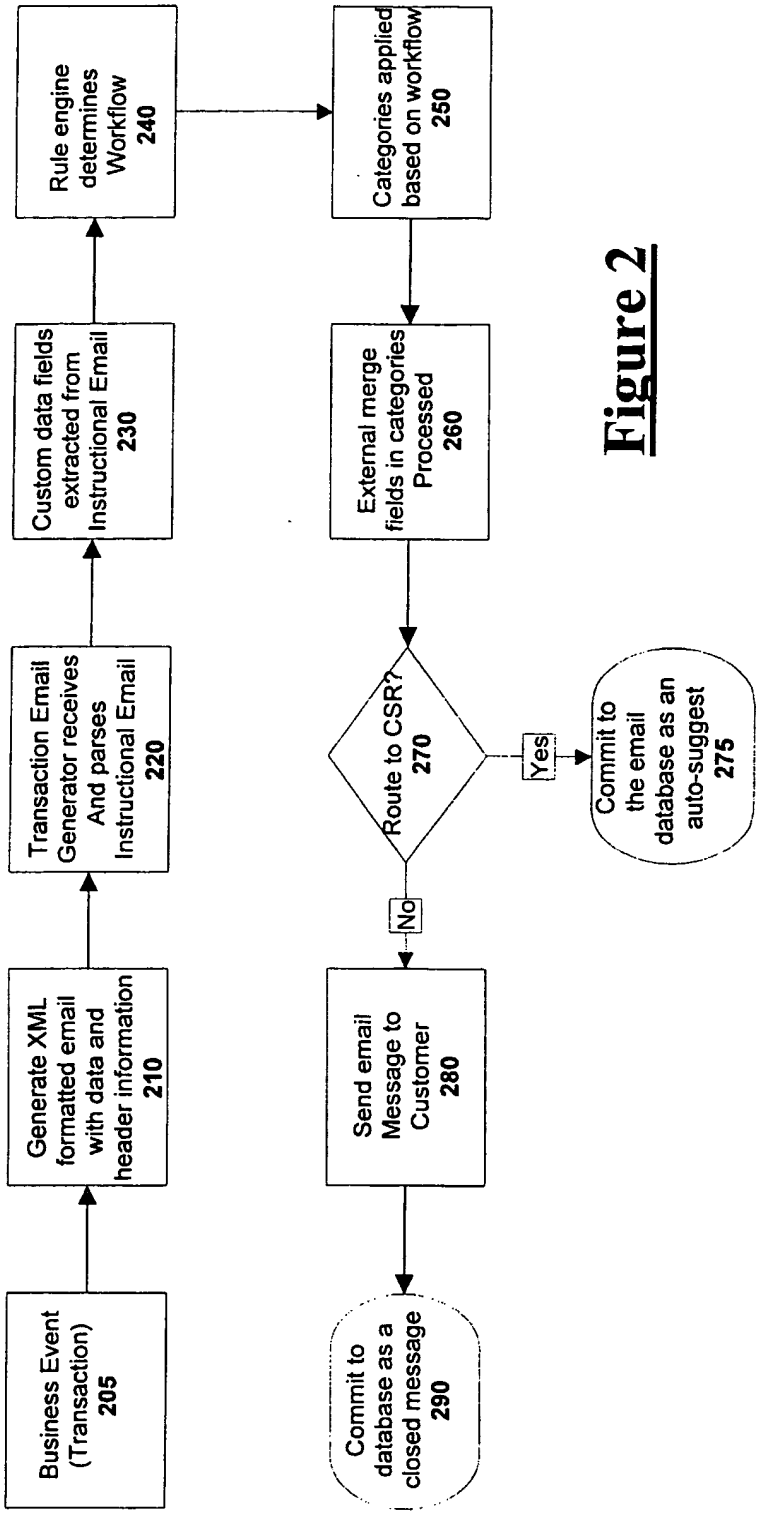

Business Event (Transaction) 205
Generate XML formatted email with data and header information 210
Transaction Email Generator receives And parses Instructional Email 220
Custom data fields extracted from Instructional Email 230
Rule engine determines Workflow 240
Categories applied based on workflow 250
External merge fields in categories Processed 260
Route to CSR? 270
Yes
Commit to the email database as an auto-suggest 275
No
Send email Message to Customer 280
Commit to database as a closed message 290

Figure 2

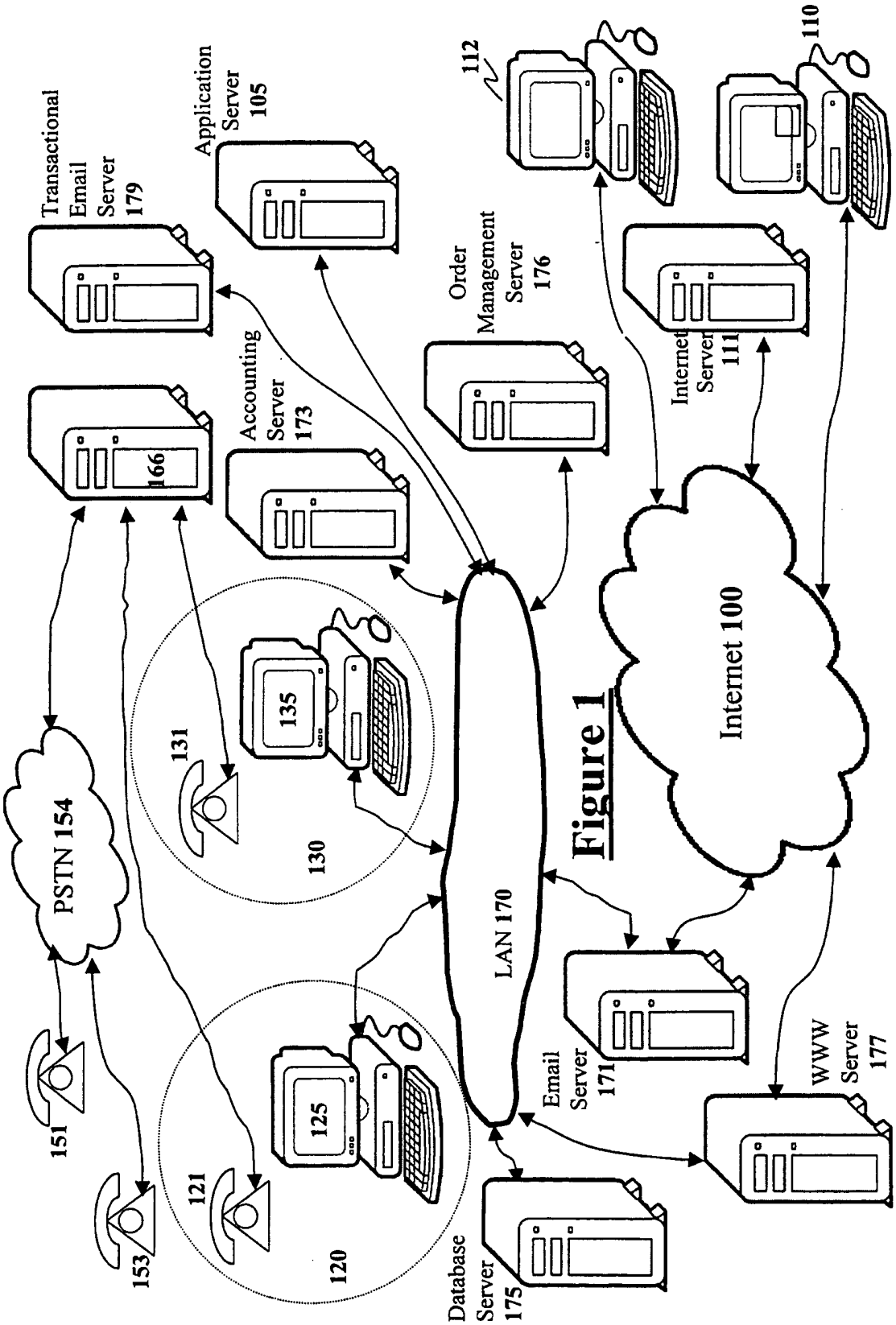

Transactional Email Server 179
Application Server 105
112
110
Order Management Server 176
Internet Server 111
166
Accounting Server 173
PSTN 154
131
135
130
LAN 170
Internet 100
151
153
121
125
120
Email Server 171
WWW Server 177
Database Server 175

Figure 1

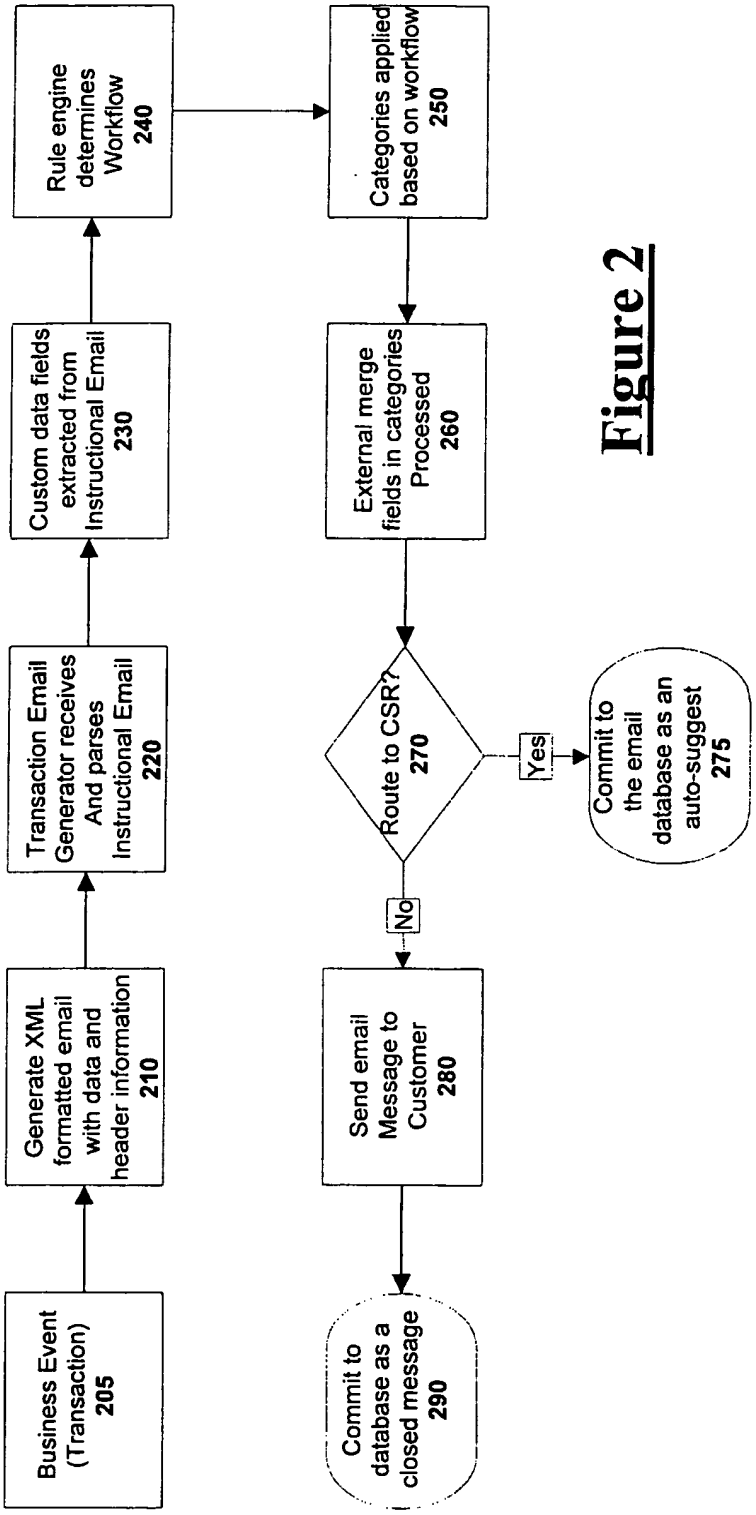

Business Event (Transaction) 205
Generate XML formatted email with data and header information 210
Transaction Email Generator receives And parses Instructional Email 220
Custom data fields extracted from Instructional Email 230
Rule engine determines Workflow 240
Categories applied based on workflow 250
External merge fields in categories Processed 260
Route to CSR? 270
Yes
Commit to the email database as an auto-suggest 275
No
Send email Message to Customer 280
Commit to database as a closed message 290

Figure 2

Figure 3A

```
From: <transactionserver@transerver.acme.com>
To: <transactions@transerver.acme.com>
Subject: Order Confirmation
Date: Mon, 21 Jun 1999 16:41:00 -0700
MIME-Version: 1.0
Content-Type: multipart/alternative;
   boundary="----=_NextPart_000_0005_01BEBC04.D8063890"
X-Priority: 3
X-MSMail-Priority: Normal
X-Mailer: Microsoft Outlook Express 4.72.3612.1700
X-MimeOLE: Produced By Microsoft MimeOLE V4.72.3612.1700
X-CustomerEvent-OrderConfirmation
X-CustomerEvent-GoldCustomer

This is a multi-part message in MIME format.

------=_NextPart_000_0005_01BEBC04.D8063890
Content-Type: text/xml;
   charset="iso-8859-1"
Content-Transfer-Encoding: quoted-printable
<?xml version="1.0"?>
<Root>
  <Message>
     <Attribute name="Message Type">OrderStatus</Attribute>
    <Attribute name="Email Address">
              joe@public.com<Attribute/>
    <Attribute name="account_number">1234567<Attribute/>
    <Attribute name="order_description">
              http://customer/order?<Attribute/>
    <Attribute name="order_number">
              http://customer/ordernumber?<Attribute/>
    <Attribute name="order_status">
              http://customer/status?<Attribute/>
</Message>
</Root>
------=_NextPart_000_0005_01BEBC04.D8063890--
```

Figure 3B

```
From: "Joe Public" <joe@public.com>
To: <transactions@transerver.acme.com>
Subject: Order Confirmation
Date: Mon, 21 Jun 1999 16:41:00 -0700
MIME-Version: 1.0
Content-Type: multipart/alternative;
  boundary="----=_NextPart_000_0005_01BEBC04.D8063890"
X-Priority: 3
X-MSMail-Priority: Normal
X-Mailer: Microsoft Outlook Express 4.72.3612.1700
X-MimeOLE: Produced By Microsoft MimeOLE V4.72.3612.1700
X-CustomerEvent-OrderConfirmation
X-CustomerEvent-GoldCustomer

This is a multi-part message in MIME format.

------=_NextPart_000_0005_01BEBC04.D8063890
Content-Type: text/xml;
  charset="iso-8859-1"
Content-Transfer-Encoding: quoted-printable
<?xml version="1.0"?>
<Root>
  <Message>
     <Attribute name="Message Type">OrderStatus</Attribute>
    <Attribute name="account_number”>1234567<Attribute/>
    <Attribute name="order_description">
          http://customer/order?<Attribute/>
    <Attribute name="order_number">
          http://customer/ordernumber?<Attribute/>
    <Attribute name="order_status">
          http://customer/status?<Attribute/>
</Message>
</Root>
------=_NextPart_000_0005_01BEBC04.D8063890--
```

Figure 4

```
Dear <customerfirstname>,

Thank you for ordering from Acme.com.  The following is a
confirmation of the order you have recently placed:

Order number
<merge url="<merge name=order_number>last=true">

Description
<merge url="<merge
name=order_description>account_number=<merge
name="account_number">">

Status
<merge url="<name=order_status>last=true">
```

Figure 5

As a gold customer of Acme.com, we would like to thank you for placing this order, and would like to present you with the opportunity to get 20% off the next item you buy from Acme.com. Just go to http://my.acme.com and save 20%.

Sincerely,

John Q. Agent
Gold Customer Service
Acme.com

Figure 6

To: Joe Public joe@public.com
From: Order Confirmation orders@acme.com
Subject: Re: Order Confirmation

Dear Joe,

Thank you for ordering from Acme.com. The following is a confirmation of the order you have recently placed:

Order number
127845

Description
Beanie Baby

Status
Backordered - expected to ship 7/31/99

Figure 7

As a gold customer of Acme.com, we would like to thank you for placing this order, and would like to present you with the opportunity to get 20% off the next item you buy from Acme.com. Just go to http://my.acme.com and save 20%.

Sincerely,

John Q. Agent
Gold Customer Service
Acme.com